**C++**

**C++ es un lenguaje de programacion, creado a mediados de 1980 por Bjarne Strousstrup, como extension del lenguaje C. Este lenguaje abarca tres paradigmas de la Programacion:**

**1. Programacion Estructurada**

**2. Programacion Generica**

**3. Programacion Orientada a Objeto**

**Un archivo Makefile es un archivo de texto en el cual se distinguen cuatro tipos basicos de declaraciones:**

**1. Comentarios: Al igual que en los programas, contribuyen a un mejor entendimiento de las reglas definidas en el archivo. Los comentarios se inician con el caracter #, y se ignora todo lo que continue despues de ella, hasta el final de linea.**

**2. Variables: Se definen utilizando el siguiente formato:**

**nombre = dato**

**De esta forma, se simplifica el uso de los archivos Makefile. Para obtener el valor se emplea la variable encerrada entre parentesis y con el caracter $ al inicio, en este caso todas las instancias de $(nombre) serian reemplazadas por datos. Por ejemplo, la siguiente definicion:**

**SRC = main.c**

**origina la siguiente linea:**

**gcc $ SRC**

**y seria interpretada como:**

**gcc main.c**

**Sin embargo, pueden contener mas de un elemento. Por ejemplo:**

**objects = programa 1.o programa 2.o programa 3.o \ programa 4.o programa 5.o**

**programa: $(objects) gcc -o programa $(objects)**

**Hay que notar que make hace distincion entre mayusculas y minusculas.**

**¿Que son las variables?**

**Variables: En programación, una variable es un espacio de memoria reservado para almacenar un valor que corresponde a un tipo de dato soportado por el lenguaje de programación.**

**Una variable es representada y usada a través de una etiqueta (un nombre) que le asigna un programador o que ya viene predefinida.**

**Es recomendable que una variable no tenga espacios, caracteres especiales (acentos o signos), y tiene que empezar con una letra.**

**C++ maneja varios tipos de variables: numéricas, de caracteres y boolean (cierto o falso) esencialmente.**

**¿Como se declaran variables en C++?**

**Para declarar una variable numérica llamada peso que inicie con un valor de 0, escribimos esto: int peso=0;**

**Para declarar una variable numérica llamada distancia que inicie con un valor de 2.67, escribimos esto: float distancia=2.67;**

**Para declarar una variable de carácter llamada inicial, con un valor de R, escribimos esto: char inicial='R';**

**Para declarar una variable boolean llamada terminado, que empiece como falso, escribimos esto: bool terminado=false;**

**Las variables que podemos declarar en C++ son las siguientes:**

**Para variables numéricas sin decimales se usa: int**

**Para variables numéricas con decimales se usa: float**

**Para variables de un solo caracter se usa: char**

**Para variables boolean se usa: bool**

**¿Puedo declarar mas de una variable en C++?**

**Solo se pone el tipo de dato (int, float, char, bool) y separamos las variables con comas.**

**int num=0, peso=16, altura=10;**

**Siempre va un punto y coma (;) al final de la declaración de variables.**

**¿Qué es un scope O "Vida" en C++?**

**Un Scope es cuánto dura una variable en memoria, además de ver en qué parte de la misma se encuentra y cómo funciona.**

**En el código siguiente:**



**En el ejemplo mostrado, la vida o scope del codigo seria desde la linea 3(int main) hasta la 4, ya que ahi termina el bloque asignado.**

**Por que motivo tienen vida los bloques?**

**Las variables tienen esa duración limitada porque C++ tiene un tiempo de vida, en el que las variables se guardan en la pila, y se eliminan cuando la pila de la función desaparece.**

**Es decir, se van creando las variables en el stack y se van sacando a medida que se van creando.**

**Es muy importante conocer bien todo sobre el Scope, ya que debemos saber hasta cuánto duran las variables para poder saber así cuándo poder utilizarlas,**

**detectando así estos errores sin perder tiempo en su depuración y agilizando el desarrollo de nuestro código.**

**Operadores Aritmeticos en C++**

**Los operadores aritmeticos basicos en c++ serian los siguientes:**



**La mayoría de estos operadores nos resultan conocidos, pues ya los hemos estudiado con el pseudocódigo. Las reglas de prioridad son las mismas que explicamos al hablar de pseudocódigo. Los paréntesis definen la prioridad principal. El siguiente orden de prioridad corresponde a las operaciones de multiplicación y división. Las operaciones de suma y resta tienen menor precedencia. Y en último lugar tendríamos la operación para obtener el módulo de una división entera.**

**C no reconoce el operador ^ para el cálculo de potencias de un número, ni el operador div o barra invertida para la obtención del cociente o división entera de una división. Existen otras alternativas que permiten realizar estas operaciones como veremos más adelante.**

**Existen otros operadores admitidos que constituyen formas de expresar abreviadamente una operación. Por ejemplo += se puede usar para indicar que la variable a la izquierda toma el valor resultante de sumarse a sí misma con la variable o expresión a la derecha. Si A=4 y se ejecuta A +=3; entonces A pasa a tomar el valor 7, equivalente a realizar la operación A = A + 3;**

**Algunos programadores utilizan con frecuencia expresiones abreviadas porque les resultan más rápidas a la hora de escribir código.**

**Operadores de Relacion en c++**

**Los operadores de relacion basicos en c++ serian los siguientes:**



**Los operadores relacionales, también denominados operadores binarios lógicos y de comparación, se utilizan para comprobar la veracidad o falsedad de determinadas propuestas de relación (en realidad se trata respuestas a preguntas). Las expresiones que los contienen se denominan expresiones relacionales. Aceptan diversos tipos de argumentos, y el resultado, que es la respuesta a la pregunta, es siempre del tipo cierto/falso, es decir, producen un resultado booleano.**

**Si la propuesta es cierta, el resultado es true (un valor distinto de cero), si es falsa será false (cero)."**

**Como puede verse, todos ellos son operadores binarios (utilizan dos operandos), de los cuales, dos de ellos son de igualdad: == y !=, y sirven para verificar la igualdad o desigualdad entre valores aritméticos o punteros. Estos dos operadores pueden comparar ciertos tipos de punteros, mientras que el resto de los operadores relacionales no pueden utilizarse con ellos.**

**Cualquiera que sea el tipo de los operandos, por definición, un operador relacional, produce un bool (true o false) como resultado, aunque en determinadas circunstancias puede producirse una conversión automática de tipo a valores int (1 si la expresión es cierta y 0 si es falsa).**

**Operadores Logicos en c++**

**Los operadores lógicos producen un resultado booleano, y sus operandos son también valores lógicos o asimilables a ellos (los valores numéricos son asimilados a cierto o falso según su valor sea cero o distinto de cero). Por el contrario, las operaciones entre bits producen valores arbitrarios."**

**Los operadores lógicos son tres, dos de ellos son binarios y el último (negación) es unario:**



**Operadores de Bits en c++**

**Los operadores de movimiento son operadores a nivel de bits, y lo que hacen es convertir una determinada cantidad a su equivalente en bits para posteriormente realizar un desplazamiento de dicho valor. Estos operadores son:**



**A pesar de ser "Operadores para manejo de bits", todos ellos exigen operandos de tipo entero, que puede ser de cualquiera de sus variantes (short, long, signed o unsigned) y enumeraciones. Es decir, el material de partida son bytes, uno o varios, dependiendo del tipo de entero utilizado.**

**Si los operandos no son enteros el compilador realiza la conversión pertinente, por lo que el resultado es siempre un entero del mismo tipo que los operandos.**

**No se debe confundir los operadores de bits, & y |, con los operadores lógicos && y ||.**